Name: **Student id:** **Sect#:** **#:**

QUESTION #	1	2	3	4	5	TOTAL
MAX POINTS	12	12	16	15	11	66
POINTS EARNED						

****

UNIVERSITY OF BAHRAIN **COLLEGE OF INFORMATION TECHNOLOGY**
DEPARTMENT OF COMPUTER SCIENCE

ITCS 242: ASSEMBLY LANGUAGE PROGRAMMING

2nd SEMESTER 2015/2016 **DATE: APR 25, 2016** **FIRST TEST**

****

QUESTION ONE: *Show your work!* *{12 points}*

1) Using 24 bits to represent signed values, the smallest decimal value is -2^{23} and the largest decimal value is $+2^{23} - 1$

2) Perform the following operation using 16's complement: 4F2A – 9E7C.

4F2A – 9E7C = 4F2A + (–9E7C)

= 4F2A + 6184 = B0AE

3) The 8-bit value 10101111 represents unsigned decimal value **175** and signed decimal value **–81**.

4) Using 8 bits to represent numbers, convert the value $(+79)_{10}$ into binary **0100 1111** and the value $(-79)_{10}$ **B1** into hexadecimal.

5) Perform the following operation: *2C7H + 3627Q*= (**A 5 E**) H.

```
0010 1100 0111
0111 1001 0111
1010 0101 1110
  A    5    E
```

6) Using 8 bits to store numbers, show how the operation $(47)_{10} – (67)_{10}$ is performed by the computer.

```
+47 = 0010 1111          0010 1111
+67 = 0100 0011    →   + 1011 1101
-67 = 1011 1101          1110 1100
```

QUESTION TWO: ***Fill in blanks*** *{12 points}*

1) The offset of the next instruction is stored in **EIP** register. The loop counter is stored in **ECX** register.
2) The flag that indicates whether the instruction result contains odd or even number of ones is **Parity**. The flag that controls the program execution (regular/single-step) is **trace**.
3) The **symbolic** addresses are translated into **logical** address by compilers/Assemblers.
4) A logical address consists of two values: **segment : offset** .
5) The Pentium processor in PCs operates in **real-address** or **protected** modes.
6) CPU registers are accessed by **names** and main memory locations are accessed by **addresses**.
7) The addressing mode used in the destination operand in the instruction: `MOV X[EBX],CX;` is **indexed.** The addressing mode used in the source operand in instruction: `SUB ECX,sizeof X;` is **immediate**.
8) In real-address mode, the CS register stores the **starting address of the active code segment** .
9) The **logical** addresses are translated into **physical** addresses by a loader.
10) In protected mode, the addresses of all program segments are stored in **the segment descriptor table** in the main memory.
11) If the physical address is **4061EH** and the segment value is **3A9F**H, then the offset value will be: **5C2E**H

    pa = seg * 10H + offset

    4061E = 3A9F * 10 + offset

    offset = (4061E - 3A9F0) / 10 = 5C2E H
12) If a computer has 8 GB of main memory, the minimum number of address lines needed is **33**.

QUESTION THREE: *Write a complete assembly program that:* *{16 points}*

- defines an array *RAT* consisting of 64 elements of signed bytes.
- fills the array *RAT* with values by randomly generating 64 values in the range ***(-25 ... +95)***
- displays in HEX all elements of array *RAT* as double words separated by a space.
- displays in signed decimal all elements of array *RAT* separated by tabs.
- stores in a variable *NEW* the sum the first, third, fifth, … bytes of array *RAT* .

```
            INCLUDE  Irvine32.inc
            .DATA
      RAT   SBYTE    64 dup(?)
      NEW   SWORD    0
            .CODE
      MAIN  PROC
            CALL     RANDOMIZE
; Generating random numbers and storing them in array RAT
            MOV      ECX, LENGTHOF TAR
            MOV      ESI, 0                ; INDEX
      L2:   MOV      EAX, 121
            CALL     RANDOMRANGE
            SUB      EAX, 25
            MOV      RAT[ESI], AL
            INC      ESI
            LOOP     L2
            CALL     CRLF
; Display elements of array RAT as dwords (HEX) separated by space
            LEA      ESI, RAT
            MOV      EBX, TYPE RAT * 4
            MOV      ECX, LENGTHOF RAT / 4
            CALL     DUMPMEM
            CALL     CRLF
; Display all elements of RAT in signed decimal separated by TABs
            MOV      ECX, LENGTHOF RAT
            MOV      ESI, 0
      L3:   MOVSX    EAX, RAT[ESI]
            CALL     WRITEINT
            MOV      AL, 9             ; ASCII OF TAB IS 9
            CALL     WRITECHAR
            INC      ESI
            LOOP     L3
; stores in NEW the sum the first, third, … bytes of array RAT
            MOV      ESI, 0
            MOV      ECX, LENGTHOF RAT / 2
      L4:   MOVSX    AX, RAT[2*ESI]
            ADD      NEW, AX
            INC      ESI
            LOOP     L4

            EXIT
      MAIN  ENDP
            END      MAIN
```

QUESTION FOUR: {15 points}

(a) Given the following data definitions, write the needed instructions to prompt the user to enter from the keyboard up to 80 characters and store them in an array named **me**. Display the entered characters at the beginning of a new line in reverse order separated by space.

```
msg     byte      "Enter up to 80 characters please: ",0
me      sbyte     81 dup (?), 0

        lea       edx, msg
        call      writestring

        mov       ecx, 81
        lea       edx, me
        call      readstring

        call      crlf
        mov       ecx, eax
        dec       ecx
L8:     mov       aL, me[ecx]
        call      writechar

        mov       aL, 32        ; space
        call      writechar
        loop      L8
```

(b) Given: `ISA  SWORD  3A78H,87CFH,...;` Write the needed instructions to store in AX the sum of high-order bytes in elements of array `ISA`.

```
        mov       ecx, lengthof  isa
        mov       esi, 0
        mov       ax, 0

L9:     movsx     bx, byte ptr isa[2*esi+1]
        add       ax, bx
        inc       esi
        loop      L9
```

QUESTION FIVE: [11 pts]

Carefully study the following Assembly code, and then answer the questions in parts: 1 and 2

```
T1    DWORD     69CB3A2CH, 248F7C39H, 725A9033H, 5C6F3A49H
T2    BYTE      5EH, 3BH, 80H, 1FH, 9AH, 22H, "ABFC6789", 9CH, 0C4H
UT    WORD      6F7FH, 6AEAH, 81CFH, 69CFH, 3459H, ?, ?, 930H, 7CH
sat   ____________________________________

      MOV       AX, WORD PTR T1+6
      MOV       BX, WORD PTR T2-2
      MOV       CH, SIZEOF UT
      MOV       CL, TYPE UT*5
      MOV       DX, WORD PTR T1
      MOVSX     DI, T2[2]
```

Part#1: Answer each of the following 3 questions as required:

1) The instruction that makes EBX point to the fourth element in UT is ***LEA EBX, UT[(4-1)*2]***
2) The instruction that swaps the last element of T1 with EBX is ***XCHG EBX,T1[sizeof T1-type T1]***
3) The instruction that stores in EAX the number of words in T2 is ***MOV EAX, SIZEOF T2/2***.
4) The statement that defines a constant sat equals to the number of bytes defined in all T1, T2, and UT directives is ***sat equ $-T1.***

Part#2: Execute the above instructions and answer each of the following 4 questions

5) The register CX will contain:
   a) 120AH b) 1018H c)1810H d) 0A12H e) None

6) The register DI will contain:
   a)006FH b)00EAH c) FFEAH d) FF6FH **e) None**

7) The register BX will contain:
   a) 493AH b)6F5CH c)3A49H **d)5C6FH** e) None

8) The register AX will contain:
   a)8F24H **b)248FH** c)8F7CH d)7C8FH e) None

Part#3: Answer each of the following questions as required:

9) The statement that produces syntax error during assembly process is:
   a)MOV AX,[EBX] **b)MOV [EBX],[EAX]** c)MOVZX EBX,ax **d)MOVZX ax,ebx** e)None

10) The statement that produces syntax error during assembly process is:
   a)INC AX b)MOVZX EBX,CL c)ADD AX,BX **d) MOVSX BX,AX** e)None

11) The statement that produces syntax error during assembly process is:
   a)MOV AX,[EBX] b)XCHG BX, CX **c)INC [EAX]** d)MOV BX,[EAX] e)None